PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 4 : G06F 1/00

A1

(11) International Publication Number: WO 88/ 05941

(43) International Publication Date: 11 August 1988 (11.08.88)

(21) International Application Number: PCT/US88/00271

(22) International Filing Date: 29 January 1988 (29.01.88)

(31) Priority Application Number: 008,855

(32) Priority Date: 30 January 1987 (30.01.87)

(33) Priority Country: US

(71) Applicant: SOFTWARE ACTIVATION, INC. [US/US]; 6 Old County Road, Lancaster, MA 01523 (US).

(72) Inventor: CHRISTENSEN, Carlos, H. ; 906 Old Road to Nine Acre Corner, Concord, MA 01742 (US).

(74) Agent: CESARI, Robert, A.; Nutter, McClennen and Fish, One International Place, Boston, MA 02110-2699 (US).

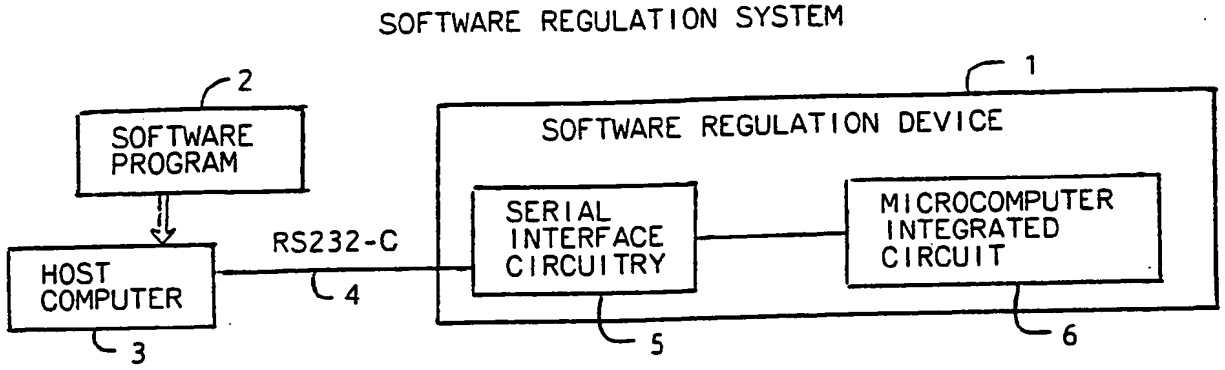
(81) Designated States: AT (European patent), AU, BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, KR, LU (European patent), NL (European patent), NO, SE (European patent), SU.

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: APPARATUS AND METHOD FOR REGULATING THE USE OF PROPRIETARY COMPUTER SOFTWARE



(57) Abstract

A software regulation system for regulating the use of a software program in a host digital data processing (computer) system. The software regulation system includes one or more checkpoint routines processed by the software program and a software regulation device, which may be part of the computer system or connected externally thereto. The checkpoint routines generate random checkpoint messages, which are enciphered and transmitted to the software regulation device. The software regulation device deciphers the checkpoint message, performs a processing operation to generate a response message, enciphers the response and sends the enciphered response to the checkpoint routine. The checkpoint routine then determines whether the enciphered response is correct and either allows the software program to proceed or terminates it.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	ML	Mali
AU	Australia	GA	Gabon	MR	Mauritania
BB	Barbados	GB	United Kingdom	MW	Malawi
BE	Belgium	HU	Hungary	NL	Netherlands
BG	Bulgaria	IT	Italy	NO	Norway
BJ	Benin	JP	Japan	RO	Romania
BR	Brazil	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	LI	Liechtenstein	SN	Senegal
CH	Switzerland	LK	Sri Lanka	SU	Soviet Union
CM	Cameroon	LU	Luxembourg	TD	Chad
DE	Germany, Federal Republic of	MC	Monaco	TG	Togo
DK	Denmark	MG	Madagascar	US	United States of America
FI	Finland				

APPARATUS AND METHOD FOR REGULATING THE USE OF PROPRIETARY COMPUTER SOFTWARE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates generally to the field of digital data processing (that is, computer) systems, and more specifically to systems for regulating the use of computer programs which run in digital data processing systems. The invention provides a system that regulates use of a computer program by means of a new regulation device, the regulated program communicating with the regulation device to determine that the program is authorized for use on the specific digital data processing system including the regulation device.

2. Description of the Prior Art

The cost of making a copy of a software program is very small in comparison to the cost of creating the original program, testing it and bringing it to market. For example, a large and complex program, which was developed and tested at great cost, can be copied onto an inexpensive floppy disk

in a few minutes using relatively inexpensive equipment such as a personal computer. In this respect, software is different from most other products, which require significant expenditures for the reverse engineering and fabrication equipment required to copy them.

Furthermore, software programs can be transported rapidly and inexpensively. For example, a program can be transmitted over a telephone line or distributed across the nation by a single satellite broadcast.

The ease with which software programs can be copied and transported contributes to the growth and prosperity of the computer industry; however, it also facilitates the unauthorized copying, distribution and use, that is, "piracy", of software programs. In fact, piracy of software occurs on a large scale and is a major problem for those who develop and sell software.

Three principal methods have been used to curb piracy, namely, legal protection, copy protection, and usage regulation. Legal protection is based on patents and copyrights, with the right to use a program based on a

license to the user. Licenses are most effective in situations in which the customer and the software publisher or distributor are in close contact. In mass market distribution of programs, that situation is less common that it once was.

Copy protection is based on technical tricks that prevent the customer from copying a software program from the disk on which it was delivered. Generally, copying was accomplished by means of various utilities in the operating system of the computer, and copy protection relied on the use of information in disk sectors that could not be obtained by the operating system but could be obtained by the program. If a copy was made using the operating system copy utilities, those disk sectors would not be copied and, when the software program looked for information in those sectors but did find it, the program terminated. Copy protection was successful for a time, but it imposed serious inconvenience on the customer and can now be defeated easily by special copying utilities and other programs.

Usage regulation depends on the use of a physically and logically secure electronic device that must be connected to

a computer before that computer runs the protected software. The software program performs instructions that test for the presence of the regulation device, and stops running if the device is not present.

Regulation devices currently in use have significant disadvantages. The typical regulation device depends on trade secrets that can be discerned relatively easily. Once those trade secrets are known, the device can be replaced by an inexpensive counterfeit device. Using a copy of the software and a counterfeit regulation device, any person can have the full benefit of the "protected" software program without purchasing the software.

Furthermore, if multiple programs are to be regulated by the same regulation device, every program developer wishing to use the device would have to know the details of how to use the device. That would require providing all of them with the details of operation of the device, which would include the trade secrets. Increasing the number of people who are aware of the trade secrets of the device increases the likelihood that the secrets of operation of the device would leak out.

In addition, since existing devices are relatively simple, they can be defeated by systematic analysis, even without the benefit of a trade secret leak. Thus current software regulation devices are effective only in certain situations and for a limited time.

Also, in some systems such as disclosed in U.S. Patent No. 4,458,315 issued to G. Uchenick on July 3, 1984, the software program does little more than detect the presence of the regulation device using a simple, easily-defeated message.

SUMMARY OF THE INVENTION

In brief summary, the invention described here is a new and improved system for regulating use of a proprietary software program including a software regulation device that is connected to a host computer running the software program to be regulated, in which the software program and software regulation device communicate so that the software program can verify that it is running on a computer that is authorized to run the software program.

The software program includes "checkpoint routines" that generate and transmit brief messages for transmission to the software regulation device over the connection between the host computer and the device, and which receives and interprets responses from the device. This exchange is a "checkpoint protocol" and has two forms, direct and indirect.

The checkpoint protocols effectively oppose attempts by an opponent, that is, one attempting to manufacture counterfeit copies of the software regulation device and distribute them with unauthorized copies of the software program, to create a counterfeit software regulation device. In a typical embodiment of the system, more than four billion different messages are possible, and they are used in an unpredictable order. Furthermore, both the challenge and the response are enciphered using fully secure cryptographic methods.

The software regulation system uses a different cryptographic key for each protected software program. This key must be kept secret, but all other aspects of the design

and operation of the software regulation device can be disclosed without weakening the protection given to the software program.

BRIEF DESCRIPTION OF THE DRAWINGS

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1 is general block diagram of a software regulation system constructed in accordance with the invention.

FIGS. 2A, 2B, 2C, and 2D are diagrams of the messages used in the software regulation system depicted in Fig. 1.

FIGS. 3 and 4 are flow charts depicting the two checkpoint routines, termed "direct" and "indirect" respectively, which are in a protected software program.

FIG. 5 is a flow chart depicting the operations in the software regulation device.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

FIG. 1 depicts one embodiment of a digital data processing system that includes the inventive software regulation system. With reference to FIG. 1, the software regulation system includes a software regulation device 1 that is connected to a host computer 3 that is running a software program 2 whose use is regulated as described below.

The connection between the host computer 3 and the software regulation device 1 is effected by means of a cable 4 over which pass serial signals conforming to the RS232 standard. The cable 4 is connected to a conventional serial port on the host computer 3 and a serial port on the software regulation device 1. It will be appreciated, however, that the mechanism for communication between the software regulation device 1 and the host computer 3 is not limited to the serial connection depicted in FIG. 1. Any communications mechanism suitable for transmitting at least

short messages, each composed of several bytes of information, is suitable for use with the invention.

The software regulation device 1 includes a microcomputer integrated circuit 6 which exchanges messages with the software program 2 through cable 4 and serial interface circuitry 5. In one specific embodiment, the microcomputer 6 is an Intel 8751 microcomputer, a member of the Intel MCS-51 family of microcontrollers manufactured by Intel Corporation. The Intel 8751 microcomputer is a single chip device, which has an on-chip read-only program memory, a read/write data memory, a CPU, and a serial port, and can execute a program contained within its program memory without requiring any references to an external memory, all of which assist in maintaining the security of the program run by the microcomputer.

In its memory, the microcomputer integrated circuit 6 contains a product registration that is associated with the program 2 running in the host computer 3. The product registration consists of at least a product number, a product key, and may also include other information about the use and status of the protected software program. In

the illustrative embodiment, only one registration is present.

In one embodiment, the product number comprises four bytes of data and the product key comprises eight bytes of data. In that embodiment, both the product number and the product key are stored in the read-only program memory of the microcomputer 6. In other embodiments of the invention, the product number and product key may be of diverse lengths, and may be stored in the read/write data memory of the microcomputer integrated circuit or in a combination of data memory and program memory. A plurality of software programs may be accommodated by a single software regulation device 1 by storing a plurality of product registrations in its memory.

An important feature of the microcomputer 6 is its integrity; specifically, it must effectively resist attempts by an opponent to examine or modify the product key. The Intel 8751 microcomputer is particularly suitable for use in the software regulation device because it permits the program memory to be "locked" so that it cannot be examined

or modified by users outside the chip without first erasing its contents.

As noted above, in the software regulation system depicted in FIG. 1, regulation is effected by means of an exchange of messages over cable 4 between the software regulation device 1 and the software program 2 running on the host computer 3. The structure of various messages is shown in FIGs. 2A through 2D.

FIG. 2A shows the structure of the message, termed herein a "challenge" message, which is first generated by a checkpoint routine in the software program 2. The challenge message consists of a four-byte padding field 11 and a four-byte random number field 12. The padding field 11 need not change from one execution of the checkpoint routine to the next, and its value is not significant; for example, the padding field could contain the ASCII codes for the four letters "ABCD". The random number field 12 is a random number or a pseudo-random number that is almost always different from one execution of a checkpoint routine to the next.

Following generation of the challenge message, and prior to transmission by the checkpoint routine, the challenge message is enciphered. FIG. 2B shows the structure of the enciphered challenge message, which is an eight-byte message that is sent by the software program through the serial port of the host computer to the software regulation device. The challenge message is enciphered using a key that is specific to the software program. This is described below as a direct checkpoint (see FIG. 3).

Alternatively, the challenge message may be enciphered using a key that is specific to a specific checkpoint. This is described below as an indirect checkpoint routine (see FIG. 4). A software program may have several checkpoint routines, each having a different key.

The enciphered challenge message is transmitted to the software regulation device 1. The software regulation device receives the enciphered challenge message, deciphers it to form the challenge message (FIG. 2A) and proceeds to shuffle it. FIG. 2C shows the structure of the shuffled challenge message. In one specific embodiment, it is obtained by exchanging the first and second halves of the

challenge. In other embodiments, other shuffling operations can be used; any shuffling operation that rearranges the contents in the challenge is sufficient. The shuffled challenge message is also enciphered, as shown in FIG. 2D into an eight-byte message. The shuffled challenge message is the message which is sent by the software regulation device 1 over cable 4 to the host computer 3. This message is the response message, and is enciphered under the same key that was used to encipher the challenge.

In the illustrative embodiment, the challenge and response messages are enciphered using the well-known Data Encryption Standard (DES) as described in Federal Information Processing Standards Publication 46, National Bureau of Standards, U. S. Department of Commerce, January 15, 1977. This enciphering method applies an eight-byte key to an eight-byte source message to produce an eight-byte enciphered message. Thus, the messages depicted in FIGs. 2A through 2D are described as being eight bytes long. If other encryption standards or mechanisms are used, the lengths of the messages may be adjusted accordingly.

The remaining three Figures, FIGS. 3, 4, and 5, describe the specific operations that are performed by a checkpoint routine in the software program 2 and by the software regulation device 1. With reference to FIG. 3, that FIG. depicts the operations performed by the software program 2 in connection with a direct checkpoint routine. This routine makes a predetermined number of attempts to perform a checkpoint protocol, that is, to generate a challenge message for transmission to the software program 2 and receive an appropriate response. If one or more attempts fail, execution of the software program may continue, but if a predetermined number of attempts fail, the routine assumes that unauthorized use is in progress and takes appropriate action, which may include termination of the software program 2.

The checkpoint routine begins by setting a trial counter to an appropriate value (step 21). This value determines how many attempts will be made to perform a successful checkpoint protocol. The choice of this value depends on circumstances which need not be directly related to software regulation, such as the possibility that the

serial port may also be connected to some other peripheral device, which may result in message errors.

Next the routine sends an unenciphered message to the software regulation device 1 (step 24) that includes an operation code field and a product number field. In the direct checkpoint routine, the contents of the operation code identify the direct checkpoint operation. The operation code for the direct checkpoint operation is the same for every direct checkpoint protocol. The product number identifies a particular software program, which will be different for different protected software programs but the same for every copy of a particular software program. In the illustrative embodiment, the operation code is one byte and the product number is four bytes.

The direct checkpoint routine then waits for the software regulation device 11 to send a ready message (step 27), which is an unenciphered message comprising a code that indicates that the software regulation device 1 is ready to continue and that it recognizes the product number. In the illustrative embodiment, the ready message is one byte long. If the ready message does not arrive within a specified

time, or if some other message is received, the checkpoint protocol fails and the routine goes to step 26.

When the ready message is received, the routine generates a challenge message (see FIG. 2A) and enciphers it to produce the enciphered challenge message 13 (see FIG. 2B) (step 28). The challenge is enciphered under a product key that is different for different protected software programs but is the same for every copy of a given program. The routine then sends the enciphered challenge message 13 to the software regulation device 11 through the serial port of the host computer 3 (step 29).

The routine then shuffles the challenge (as described above in the discussion of FIG. 2C) and uses the product key to produce the enciphered shuffled challenge (as described above in the discussion of FIG. 2D) (step 30). The result is the expected response to the challenge, and the routine saves the result for use (as described below in connection with step 32). The routine then waits for a response from the software regulation device 11 (step 31). If the response does not arrive within a specified amount of time after the enciphered challenge message was transmitted to

the software regulation device, the checkpoint protocol fails and the routine goes to step 26.

If the response message arrives, the routine compares it to the enciphered shuffled challenge which was computed in step 30 with the response (step 32). If they are the same, the checkpoint succeeds; otherwise, the checkpoint protocol fails, and the routine goes to step 26. If the checkpoint succeeds, the routine is complete and execution of the software program continues (step 33).

If the checkpoint protocol fails in any of steps 27, 31 or 32, the routine delays for a time that lets the software regulation device 1 reach its initial state (step 26). Then the routine decreases the trial count by one (step 23) and compares it to zero (step 22). If the trial count is zero, the checkpoint has failed, and the routine assumes that use of the software program is unauthorized and takes appropriate action (step 25). If the trial count is not zero, the routine sequences to step 24, where it starts a new checkpoint protocol.

FIG. 4 depicts the operations performed by an indirect checkpoint routine. This checkpoint is similar to that for a direct checkpoint (FIG. 3), but differs in three respects, all relating to the fact that the indirect checkpoint routine transmits a deciphering key which is used by the software regulation device 1 to decipher the enciphered challenge message. The software regulation device 1 uses a deciphering key identified by the product number initially transmitted by the indirect checkpoint routine (in step 44) to decipher the deciphering key. A software program 2 may have several checkpoint routines, and each indirect checkpoint routine has its own key. The operation code transmitted in step 24 (FIG. 3) of the direct checkpoint routine (and step 44 of the indirect checkpoint routine) identifies whether the routine is a direct checkpoint routine or an indirect checkpoint routine.

With reference to FIG. 4, the first difference between the direct checkpoint routine and the indirect checkpoint routine is in step 48, which follows step 47. Step 47 is equivalent to step 27 in the direct checkpoint routine depicted in FIG. 3. Following step 47, if the ready message is received from the software regulation device, the

indirect checkpoint routine sends an enciphered checkpoint key to the software regulation device 1 (step 48). That checkpoint key is enciphered under the product key. Although each protected software program has just one product key, a product that uses indirect checkpoints has a checkpoint key for each checkpoint.

The second difference is in step 50, which corresponds to step 28, in which the enciphered challenge message is generated. In step 50, the routine uses the checkpoint key which was transmitted (in enciphered form) in step 48 to encipher the challenge message, rather than the product key which is used to encipher the challenge message in step 28 of the direct checkpoint routine.

The third difference between the direct checkpoint routine and the indirect checkpoint routine is in step 52, which corresponds to step 30 in the direct checkpoint routine. In the indirect checkpoint routine the checkpoint key is used rather than the product key to encipher the shuffled challenge.

Although the indirect checkpoint routine sends a copy of the checkpoint key enciphered under the product key, it does not have to perform that enciphering. Instead, the enciphered checkpoint key is stored as part of the routine. Therefore the product key is not present in the protected software program. This is an important advantage of the indirect checkpoint (FIG. 4) over the direct checkpoint (FIG. 3). When indirect checkpoint routines are used, an opponent must find every checkpoint routine in the program in order to obtain the keys (that is, the checkpoint keys) that are necessary to construct a counterfeit software regulation device. On the other hand, when a direct checkpoint is used, the opponent need only find one checkpoint in order to find the necessary key (the product key) to be able to produce a counterfeit software regulation device 1. It will be appreciated that a software program 2 may include any combination of direct and indirect checkpoint routines.

FIG. 5 depicts a flow chart detailing the operations performed by the software regulation device 1. This routine responds to enciphered messages from both a direct checkpoint routine and an indirect checkpoint routine.

In its initial state, the software regulation device 1 is waiting for a message from a checkpoint routine comprising an operation code and a product number (step 60). When the appropriate message is received, the routine compares the product registration number in the message with product registrations that it maintains (step 61). If the software regulation device 1 contains a product registration whose product number is identical to the received product number, the device selects that registration and proceeds to step 62; otherwise, it sequences back to step 60.

In step 62, the routine examines the operation code sent by the host (step 62) to determine whether the checkpoint routine run by the software program 2 (FIG. 1) is a direct checkpoint routine (FIG. 3), an indirect checkpoint routine (FIG. 4), or another operation. If the operation code specifies a direct checkpoint, the software regulation device 1 sequences to 64. In step 64, the software regulation device 1 selects as its encryption key the product key associated with the product registration from the message, after which it sequences to step 67.

On the other hand, if the operation code specifies an indirect checkpoint, the software regulation device 1 sequences to 65. If the software regulation device 1 does not receive the message containing the enciphered checkpoint key within a predetermined amount of time, it returns to step 60. If the message is properly received, the software regulation device 1 uses the product key corresponding to the product identification sent with the operation code to decipher the checkpoint key (step 66). The deciphered checkpoint key is the key that is used to decipher the enciphered challenge message.

Following either step 64 or step 66, the software regulation device 1 sequences to step 67, in which it waits for the enciphered challenge message. If that message is not received within a predetermined time, the software regulation device 1 returns to step 60. If the message is properly received, the software regulation device 1 deciphers the enciphered challenge message using the key selected in step 64 or step 66, shuffles it and enciphers the result (step 68). The resulting message is then sent back to the host computer 3 over cable 4 (step 68).

The operation code received in step 62 may also specify other operations not specified herein. If it does, the software regulation device 1 sequences to step 63 to perform the operation, after which it returns to step 60.

As mentioned above, in one embodiment of the invention, the challenge messages include random or pseudo-random numbers. It will be appreciated that the numbers need not be truly random. It is desirable, however, that they comprise a sequence which is not repeated in patterns which could be detected by analysis of the sequence within a reasonable amount of time. In the illustrative embodiment, more than four billion different numbers are possible in the four-byte random number field, and so it is preferable that the numbers that are generated for the random number field be a significant portion of the set of numbers that are available. Indeed, a sequence of integers from zero up to four billion is a suitable sequence for use with the invention.

Furthermore, it will be appreciated by one skilled in the art that, the software program 2, instead of generating an enciphered shuffled challenge message, after sending the

enciphered challenge message, for comparison with the message received from the software regulation device 1 (see steps 30, FIG. 3, and 52, FIG. 4), may, after receiving the enciphered shuffled challenge message from the software regulation device, decipher that message and un-shuffle it and compare the result to the original challenge message. That would, however, delay verification of the success of the checkpoint, since that must be done after receipt of the enciphered shuffled challenge message from the software regulation device 1.

In addition, while the software regulation system has been described as including a software regulation device 1 separate and distinct from the host computer, it will be appreciated that the software regulation device may be part of the host computer, as long as the checkpoint routines depicted in FIGs. 3 and 4 can properly communicate with the software regulation device 1. The use of an external software regulation device facilitates portability of the authority to operate the regulated software program from one host computer to another, since the external device 1 can be easily removed from one host computer and attached to another.

It will further be appreciated that an opponent who wishes to make unauthorized use of a protected software program can undermine the system in two ways. First, he can attempt to remove all checkpoint routines from the protected software program, in which case the software regulation device 1 cannot prevent unauthorized use of the resulting program. Second, he can attempt to manufacture a counterfeit software regulation device. This invention complicates these efforts for the following reasons.

First, a direct checkpoint protocol uses a very large number of challenge and response message pairs (more than four billion in the illustrative embodiment). By using a large random number in forming the challenge message, the messages are not repeated in short, readily discernible sequences. Thus an opponent cannot construct a counterfeit software regulation device 1 that contains a complete table of challenge-response pairs.

Second, a direct checkpoint protocol enciphers both the challenge message from the software program 2 and the response message from the software regulation device 1. The

messages are enciphered under a key, the product key, that is unique to each protected software program. Thus, without knowledge of the product key, an opponent cannot build device that generates the enciphered response from the enciphered challenge, even though the relationship between the challenge messages and the response messages may be known.

Furthermore, the invention does not depend on trade secrets. The only thing that must be kept secret is the product key, which can be different for each protected software program. Attempts to obtain the product key are discouraged by keeping the key in a memory that is "locked" by the physical and logical integrity of the microcomputer 6 in the software regulation device 1. Thus an opponent cannot obtain the product key by examining the software regulation device.

The use of indirect checkpoints rather than direct checkpoints provides further security. If indirect checkpoints are used, the product key is not present anywhere in the software program. In order to make a counterfeit software regulation device, an opponent would

have to find and remove every indirect checkpoint routine in a protected software program.

The foregoing description has been limited to a specific embodiment of this invention. It will be apparent, however, that variations and modifications may be made to the invention, with the attainment of some or all of the advantages of the invention. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

What is claimed as new and desired to be secured by Letters Patent of the United States is:

CLAIMS

1. A software regulation system for regulating use of a software program operating in a host computer, the software program including a checkpoint routine and the host computer including a software regulation device;

A. said checkpoint routine including:

i. checkpoint message generation means for generating a checkpoint message;

ii. checkpoint message transmission means connected to said message generation means for enabling checkpoint messages generated by said message generation means to be transmitted by said host computer to said software regulation device;

iii. response message receiving means connected to said message transmission means for receiving response messages generated by said software regulation device following transmission of a checkpoint message; and

iv. correspondence means connected to said message generation means and said message receiving means for determining the correspondence between said checkpoint message and said received response message, said

correspondence means controlling the operation of said software program in response to that determination;

B. said software regulation device comprising:

i. checkpoint message receiving means for receiving from said checkpoint routine said checkpoint message;

ii. response message processing means connected to said checkpoint message receiving means for processing said received checkpoint message received by said checkpoint message receiving means to generate a response message therefrom; and

iii. response message transmitting means connected to said response message processing means for transmitting said response message generated by said response message processing means to said checkpoint routine.

FIG. 1

SOFTWARE REGULATION SYSTEM

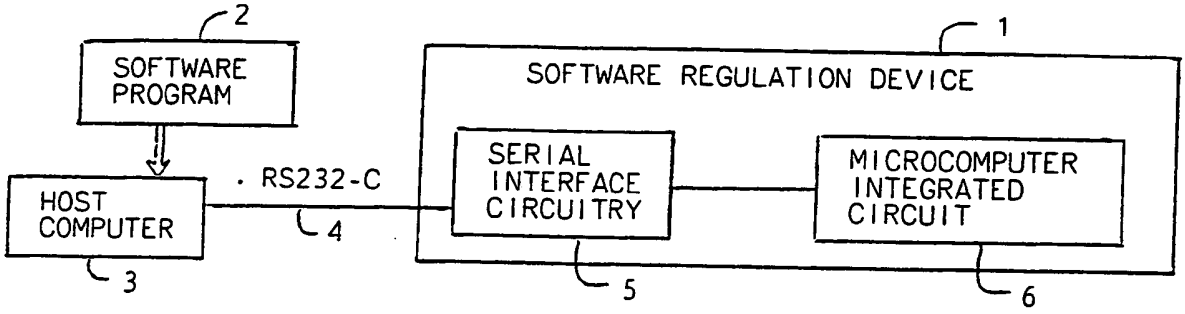


FIG. 2A

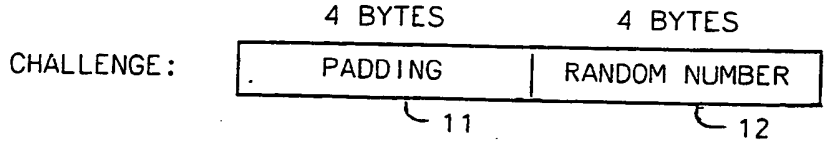


FIG. 2B

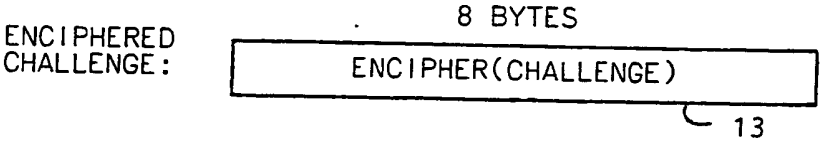


FIG. 2C

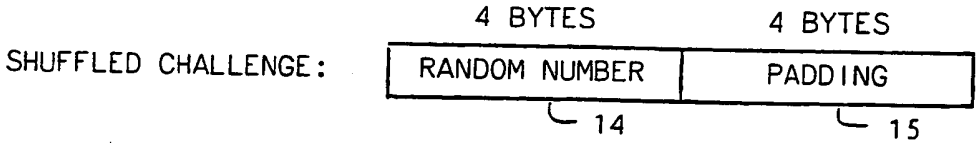


FIG. 2D

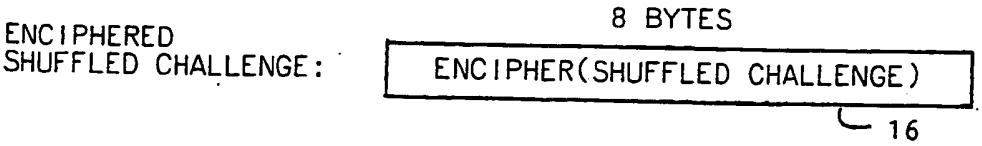


FIG. 3

DIRECT CHECKPOINT ROUTINE

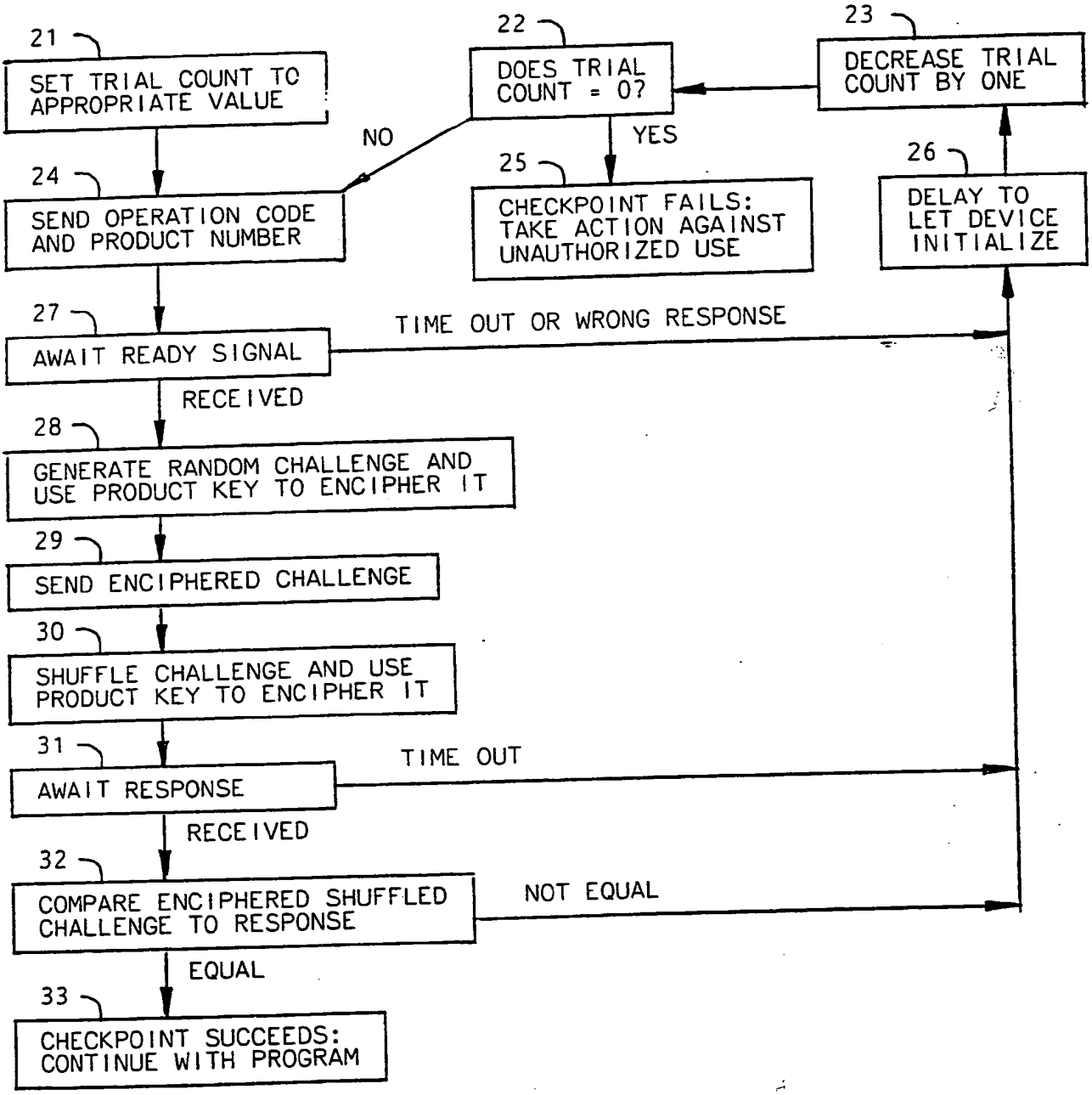


FIG. 4 3/4

INDIRECT CHECKPOINT ROUTINE

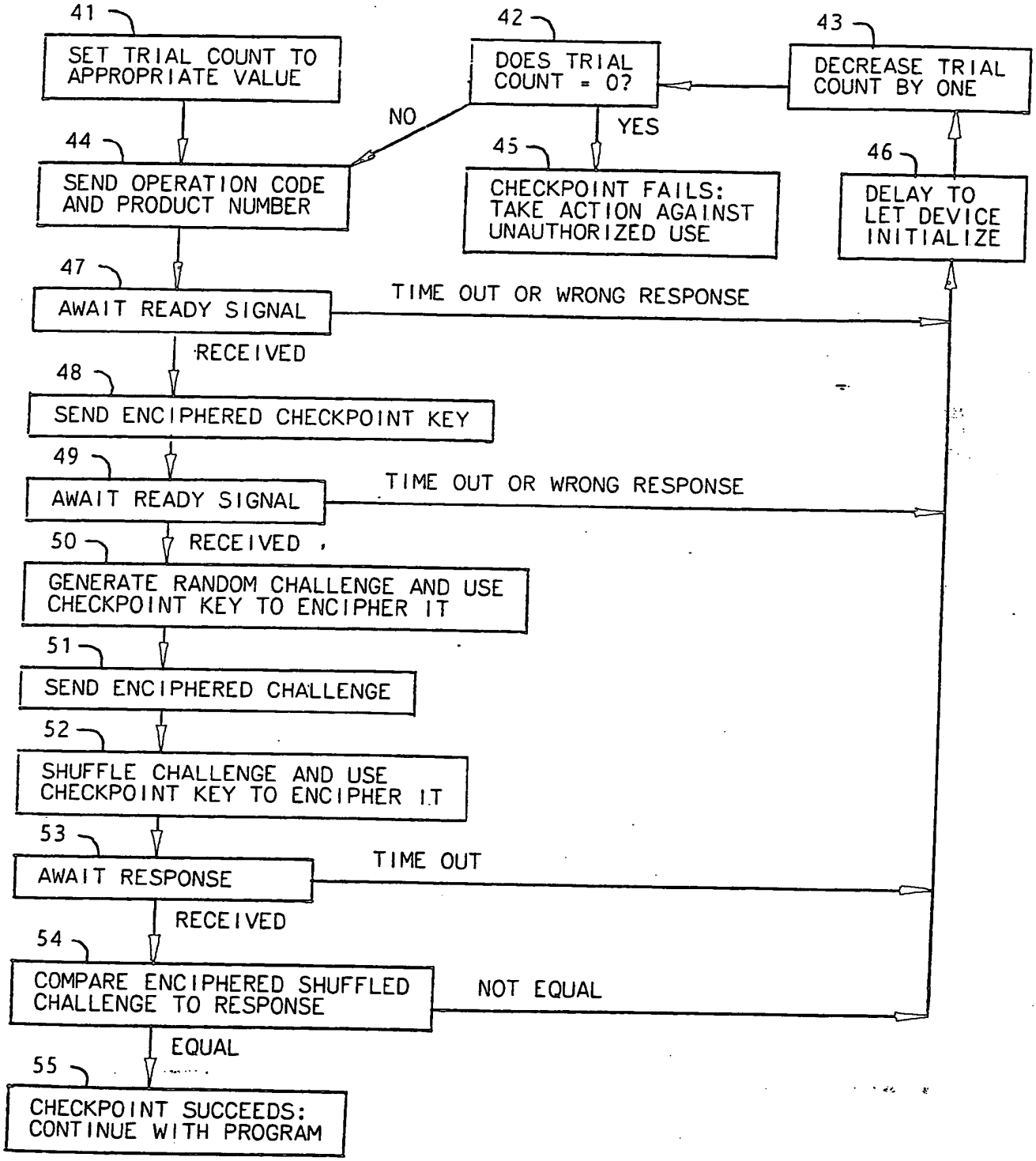
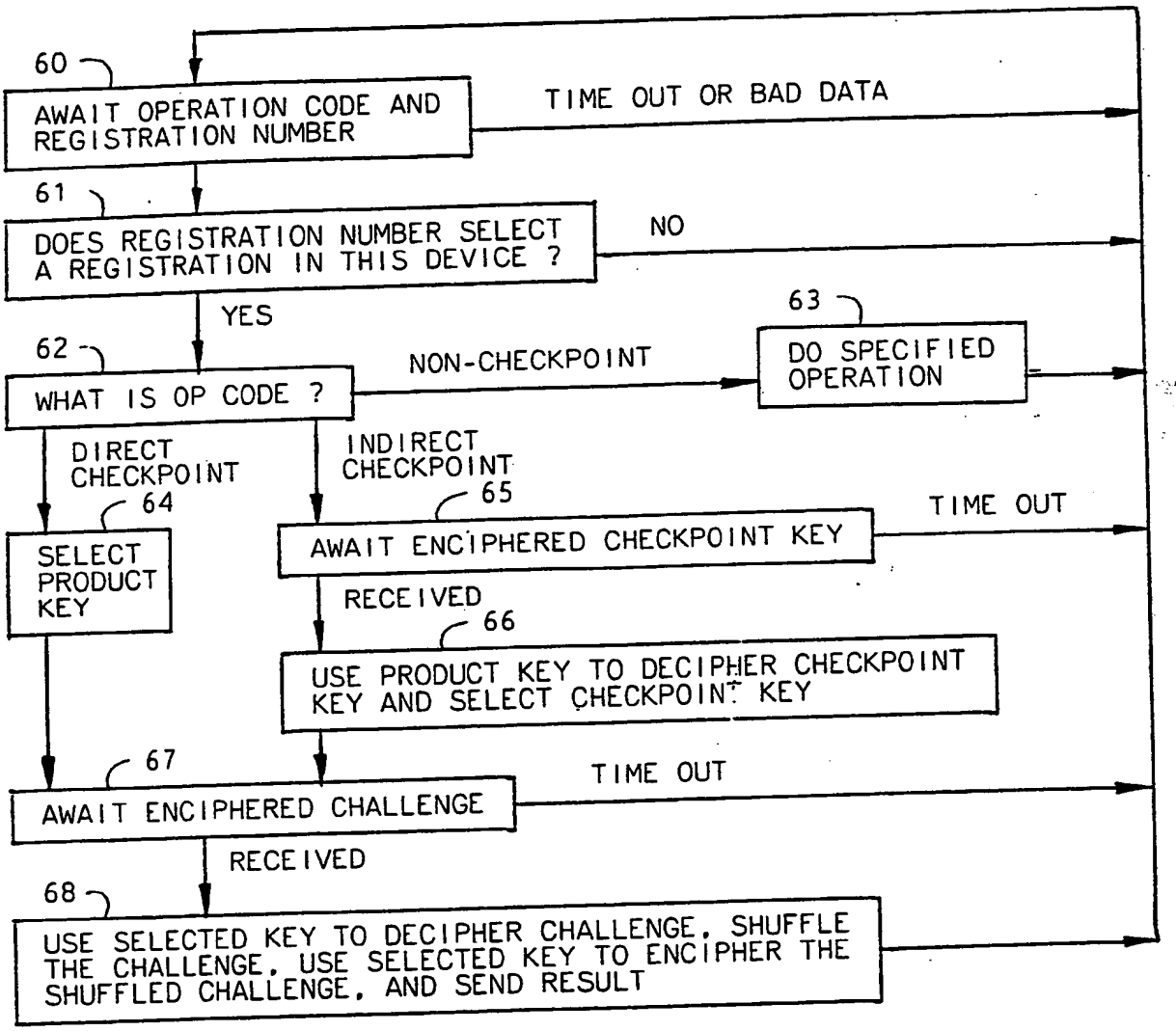


FIG. 5

DEVICE CHECKPOINT ROUTINE



INTERNATIONAL SEARCH REPORT

International Application No PCT/US 88/00271

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC[4]: G 06 F 1/00

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
IPC[4]	G 06 F 1/00

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X	EP, A, 0135422 (LEVEQUE) 27 March 1985, see page 2, line 30 - page 3, lines 3, 14-29; page 4, lines 5-19; figure 1 --	1
X	FR, A, 2577332 (LEVEQUE) 14 August 1986, see the whole document --	1
A	EP, A, 0084441 (ROGERS et al.) 27 July 1983, see abstract; page 9, lines 7-19; figure 1 --	1
A	EP, A, 0183608 (DOLLHOFF et al.) 4 June 1986, see abstract; page 7, line 22 - page 8, line 24; figures 1,2,4 --	1
A	DE, A, 3149279 (MULDER) 23 June 1983, see page 10, line 23 - page 11, line 15; page 15, lines 9-22 --	1
A	Communications of the Association of Computing Machinery, vol. 27, no. 9, September 1984 (New York, US), T. Maude et al.: "Hardware protection against software piracy", pages 950-959,	1

./.

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
7th June 1988	11 JUIL. 1988
International Searching Authority	**Signature of Authorized Officer**
EUROPEAN PATENT OFFICE	P.C.G. VAN DER PUTTEN

Form PCT/ISA/210 (second sheet) (January 1985)

International Application No. PCT/US 88/00271

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

Category *	Citation of Document, with indication, where appropriate, of the relevant passages	Relevant to Claim No
	see page 951, right-hand column, line 48 - page 952, right-hand column, line 7	

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO. US 8800271 SA 21162

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report. The members are as contained in the European Patent Office EDP file on 28/06/88
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A- 0135422	27-03-85	FR-A- 2550638	15-02-85
FR-A- 2577332	14-08-86	None	
EP-A- 0084441	27-07-83	None	
EP-A- 0183608	04-06-86	JP-A- 61175729	07-08-86
DE-A- 3149279	23-06-83	None	